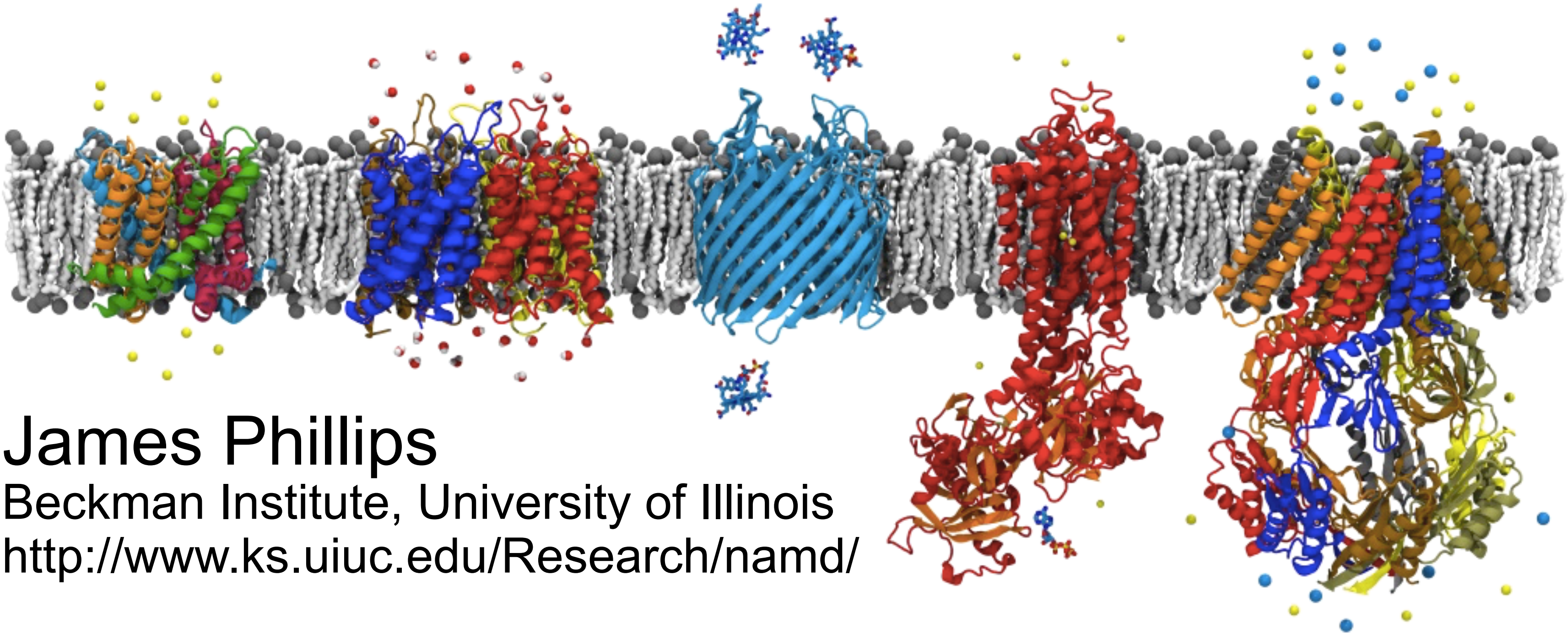
Lessons in Portability from NAMD and Charm++

SC15 Workshop on Portability Among HPC Architectures for Scientific Applications



James Phillips

Beckman Institute, University of Illinois

http://www.ks.uiuc.edu/Research/namd/

NIH Biomedical Technology Research Center for Macromolecular Modeling and Bioinformatics

Developers of the widely used computational biology software VMD and NAMD

290,000 registered VMD users
72,000 registered NAMD users

Renewed 2012-2017 with 10.0 score (NIH)

research projects include: virus capsids, ribosome, photosynthesis, protein folding, membrane reshaping, animal magnetoreception

600 publications (since 1972)
over 54,000 citations

5 faculty members
8 developers
1 systems administrator
17 postdocs
46 graduate students
3 administrative staff

Achievements Built on People

Tajkorshid, Luthey-Schulten, Stone, Schulten, Phillips, Kale, Mallon

nature

THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE

THE HIV-1 CAPSID

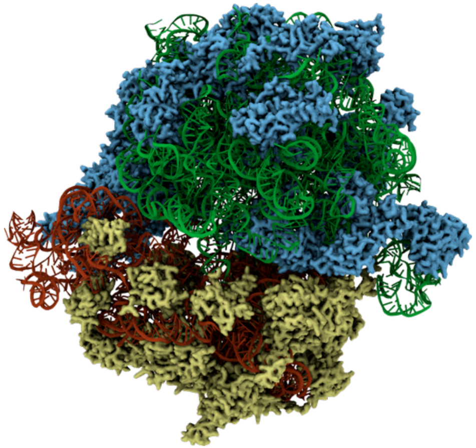
Atomic structure of the AIDS pathogen's protein coat

PAGE 643

2013 *HPCwire* Editors' Choice Award for Best Use of HPC in Life Sciences

Other Petascale Projects Using NAMD

From cellular machines
to the pharmacy...



ribosome

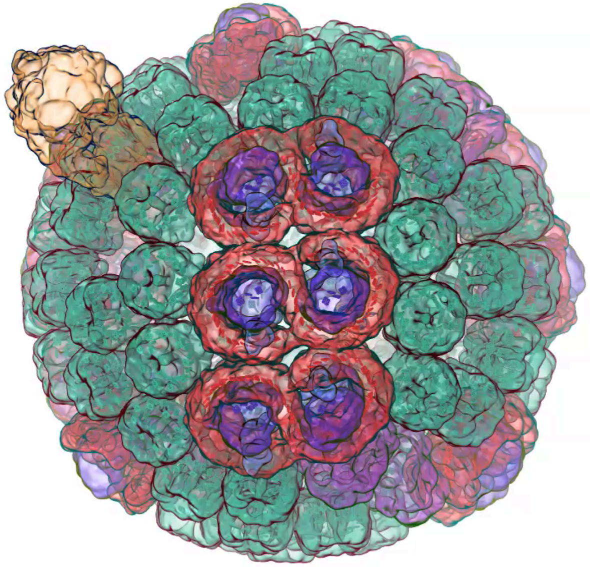
3 M atoms, multiple copies

From woodchips to gasoline...

second-generation biofuels

> 10 M atoms

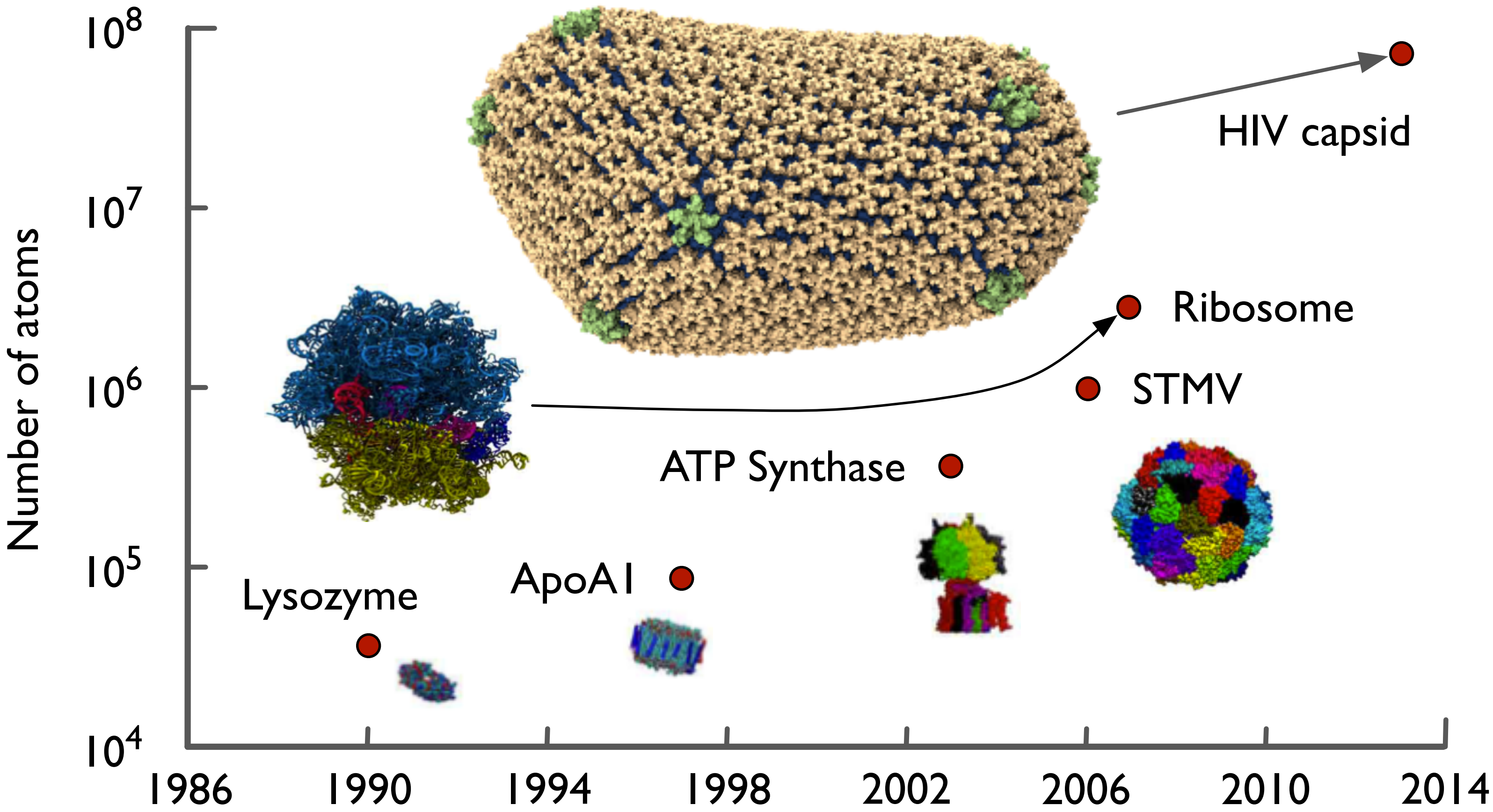
From solar energy to cellular fuel...



photosynthetic
chromatophore

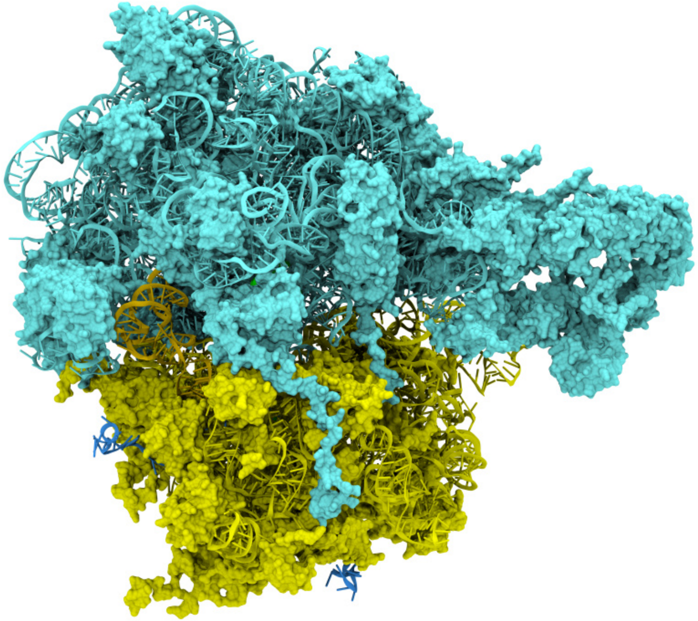
100 M atoms

A brief history of NAMD (and VMD)

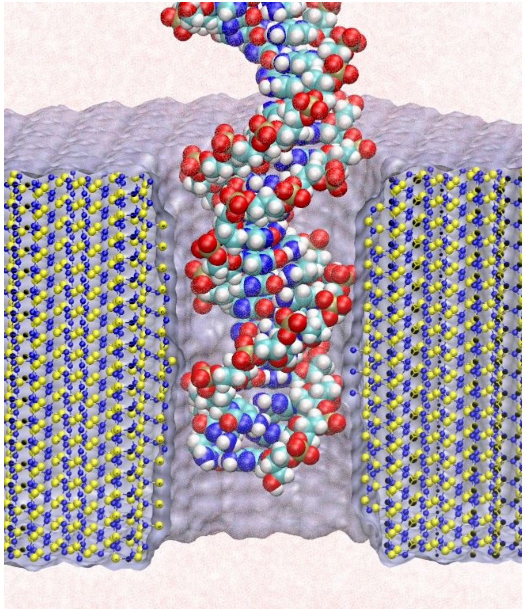


Computational Microscopy

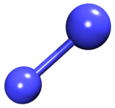
Ribosome: synthesizes proteins from genetic information, target for antibiotics

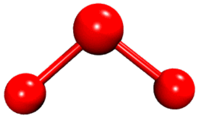


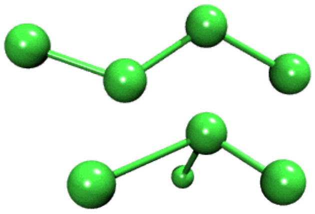
Silicon nanopore: bionanodevice for sequencing DNA efficiently



Molecular Mechanics Force Field







$$U(\vec{R}) = \underbrace{\sum_{bonds} k_i^{bond}(r_i - r_0)^2}_{U_{bond}} + \underbrace{\sum_{angles} k_i^{angle}(\theta_i - \theta_0)^2}_{U_{angle}} +$$

$$\underbrace{\sum_{dihedrals} k_i^{dihe}[1 + \cos(n_i\phi_i + \delta_i)]}_{U_{dihedral}} +$$

$$\underbrace{\sum_i \sum_{j \neq i} 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] + \sum_i \sum_{j \neq i} \frac{q_i q_j}{\epsilon r_{ij}}}_{U_{nonbond}}$$

Classical Molecular Dynamics

Energy function: $U(\vec{r}_1, \vec{r}_2, \cdots \vec{r}_N) = U(\vec{R})$

used to determine the force on each atom: $m_i \frac{d^2 \vec{r}_i}{dt^2} = \vec{F}_i = -\vec{\nabla} U(\vec{R})$

Newton's equation represents a set of N second order differential equations which are solved numerically via the Verlet integrator at discrete time steps to determine the trajectory of each atom.

$$\vec{r}_i(t + \Delta t) = 2\vec{r}_i(t) - \vec{r}_i(t - \Delta t) + \frac{\Delta t^2}{m_i} \vec{F}_i(t)$$

Small terms added to control temperature and pressure.

Long-term Charm++ Collaboration

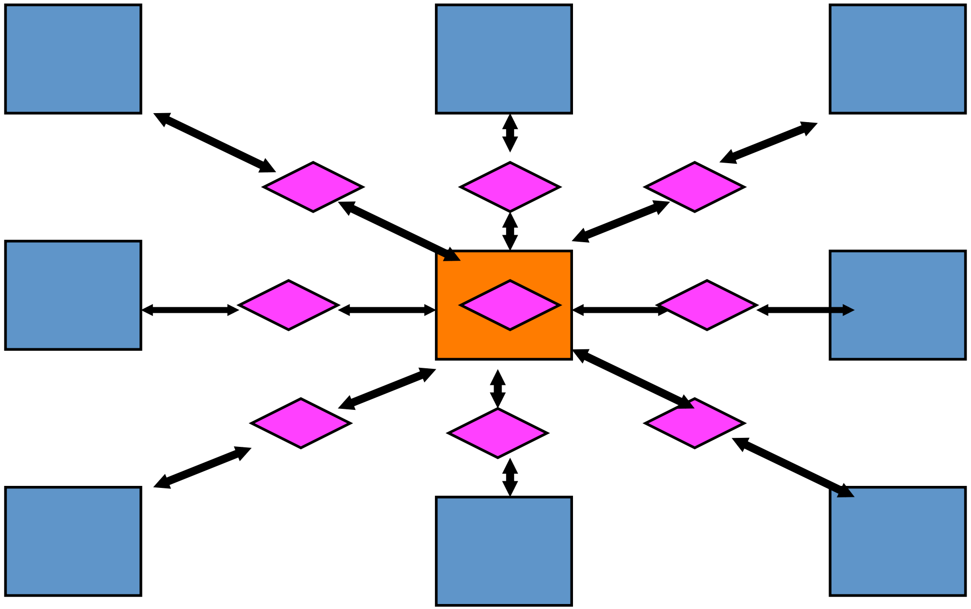
- Illinois Parallel Programming Lab
  - Prof. Laxmikant Kale
  - charm.cs.illinois.edu

KLAUS SCHULTEN
Center for the Physics of Living Cells
University of Illinois
2012 IEEE COMPUTER SOCIETY
SIDNEY FERNBACH
MEMORIAL AWARD
LAXMIKANT V. KALE
National Center for Supercomputing
Applications · University of Illinois

- Long standing collaboration
  - Since start of Center in 1992
  - Gordon Bell award at SC2002
  - Joint Fernbach award at SC12
- Synergistic research
  - NAMD requirements drive and validate CS work
  - Charm++ software provides unique capabilities
  - Enhances NAMD performance and flexibility

NAMD 2 Hybrid Decomposition

Kale *et al., J. Comp. Phys.* 151:283-312, 1999.



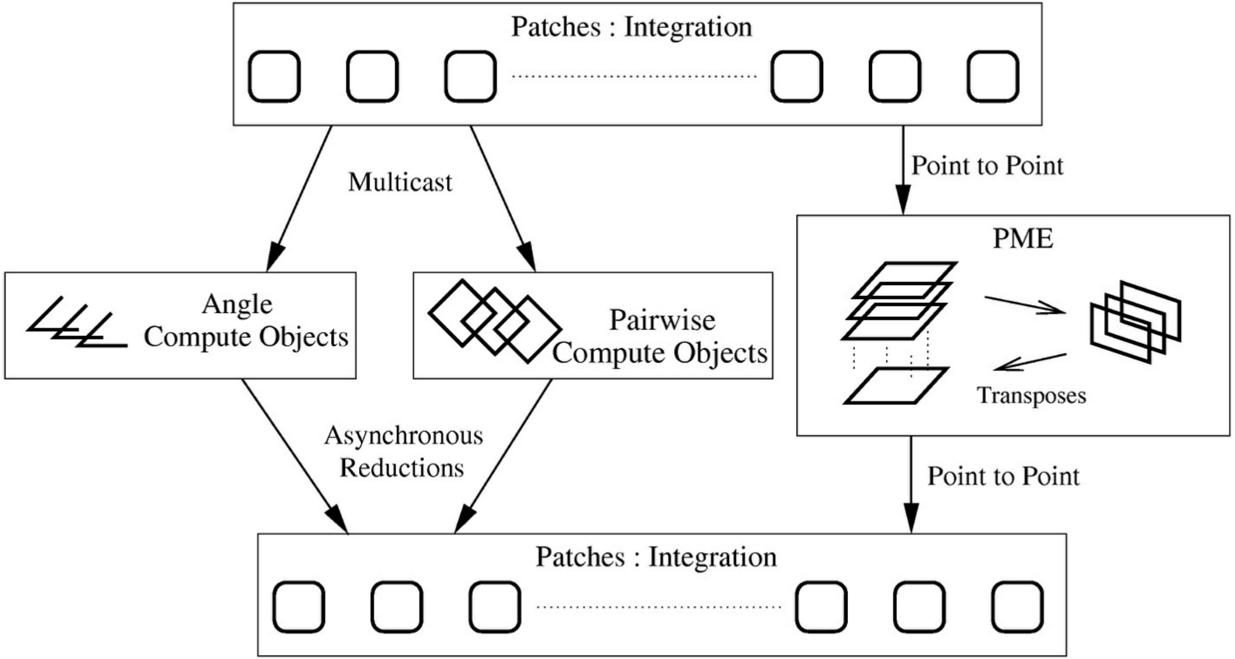
- Spatially decompose data and communication.
- Separate but related work decomposition.
- "Compute objects" facilitate iterative, measurement-based load balancing system.

Implemention in 1997 Charm++

- Parallel C++ with *data driven* objects.
- Object groups:
  - Global object with a “representative” on each PE.
- Asynchronous method invocation.
- Prioritized scheduling of messages/execution.
- Measurement-based load balancing.
- Portable messaging layer.

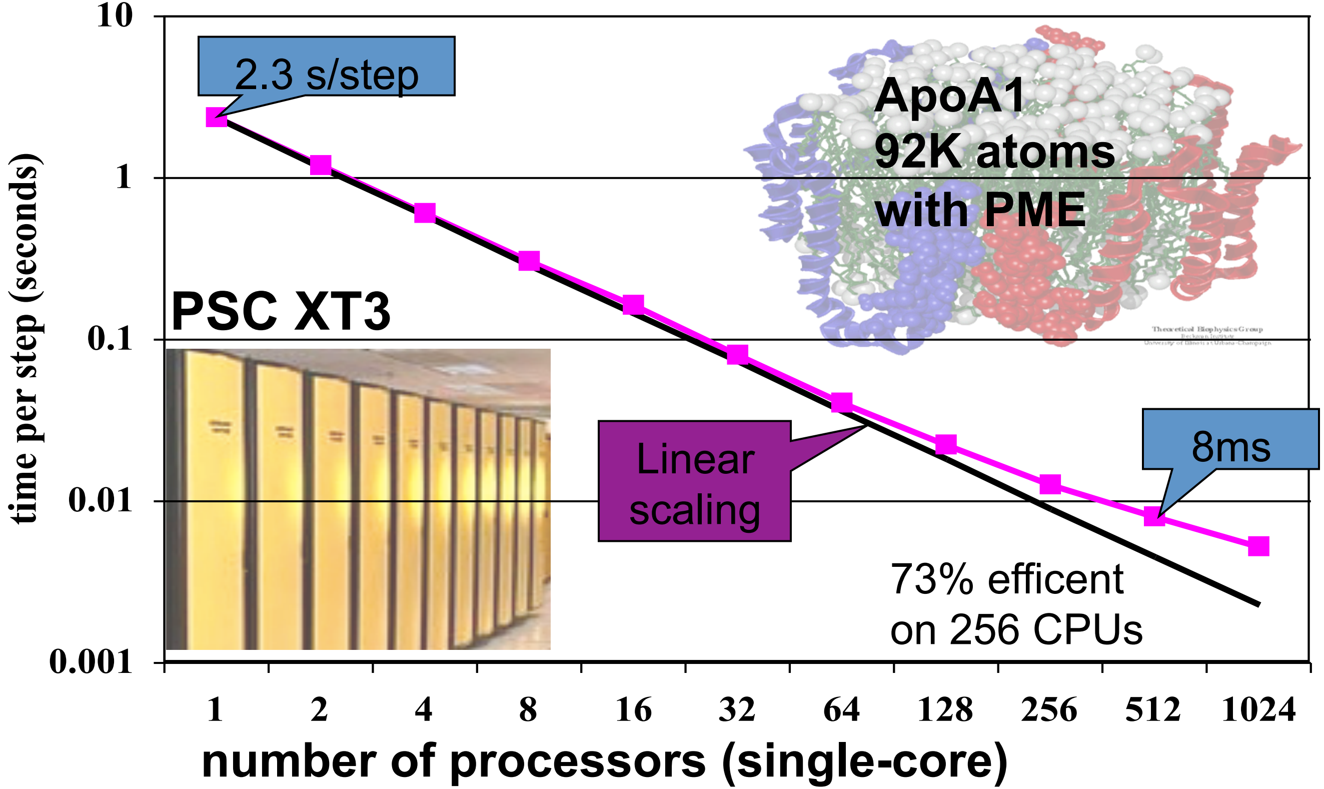
NAMD Overlapping Execution

Phillips *et al., SC2002*.



Objects are assigned to processors and queued as data arrives.

2006 NAMD Performance



2.3 s/step

ApoA1
92K atoms
with PME

PSC XT3

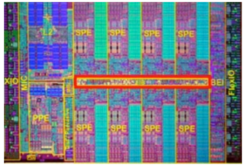
Linear scaling

8ms

73% efficent
on 256 CPUs

Early Acceleration Options

- Outlook in 2005-2006:
  - FPGA reconfigurable computing (with NCSA)
    - Difficult to program, slow floating point, expensive



  - Cell processor (NCSA hardware)
    - Relatively easy to program, expensive
  - ClearSpeed (direct contact with company)
    - Limited memory and memory bandwidth, expensive

  - MDGRAPE
    - Inflexible and expensive

  - Graphics processor (GPU)
    - Program must be expressed as graphics operations

CUDA: Practical Performance

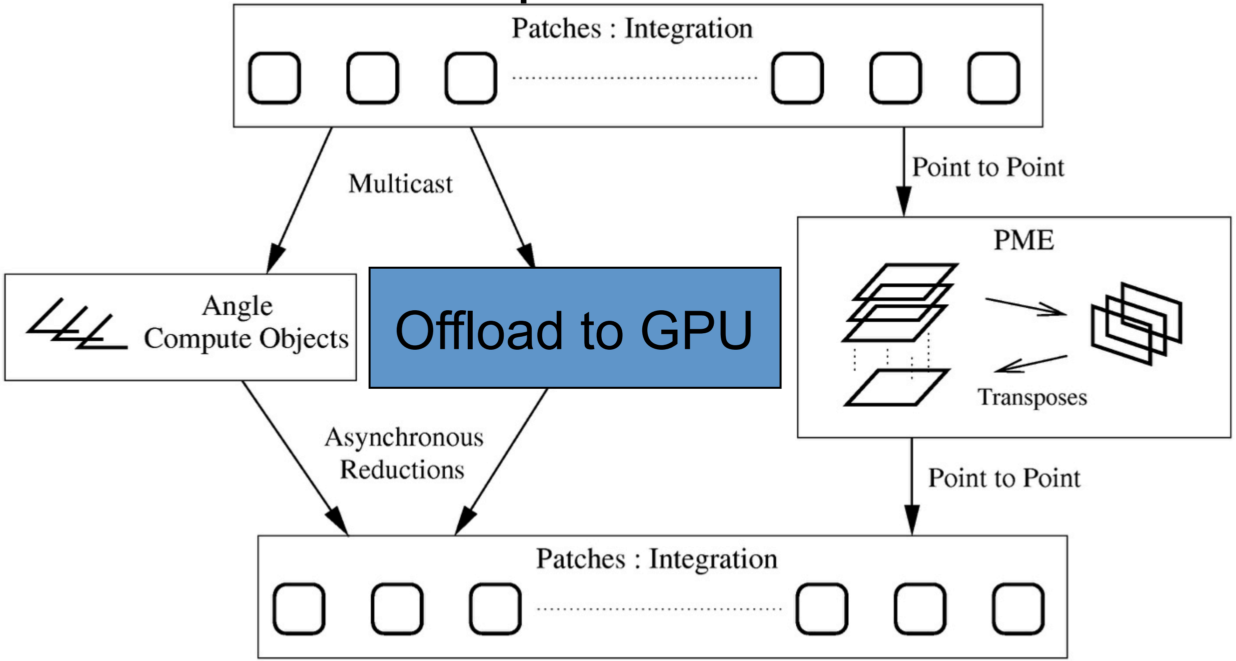
November 2006: NVIDIA announces CUDA for G80 GPU.

- CUDA makes GPU acceleration usable:
  - Developed and supported by NVIDIA.
  - No masquerading as graphics rendering.
  - New shared memory and synchronization.
  - No OpenGL or display device hassles.
  - Multiple processes per card (or vice versa).
- Center and collaborators make it useful:
  - Experience from VMD development
  - David Kirk (Chief Scientist, NVIDIA)
  - Wen-mei Hwu (ECE Professor, UIUC)

Stone *et al.*, *J. Comp. Chem.* 28:2618-2640, 2007.

NAMD Overlapping Execution

Phillips *et al., SC2002*.



Objects are assigned to processors and queued as data arrives.

Gearing Up for Petascale

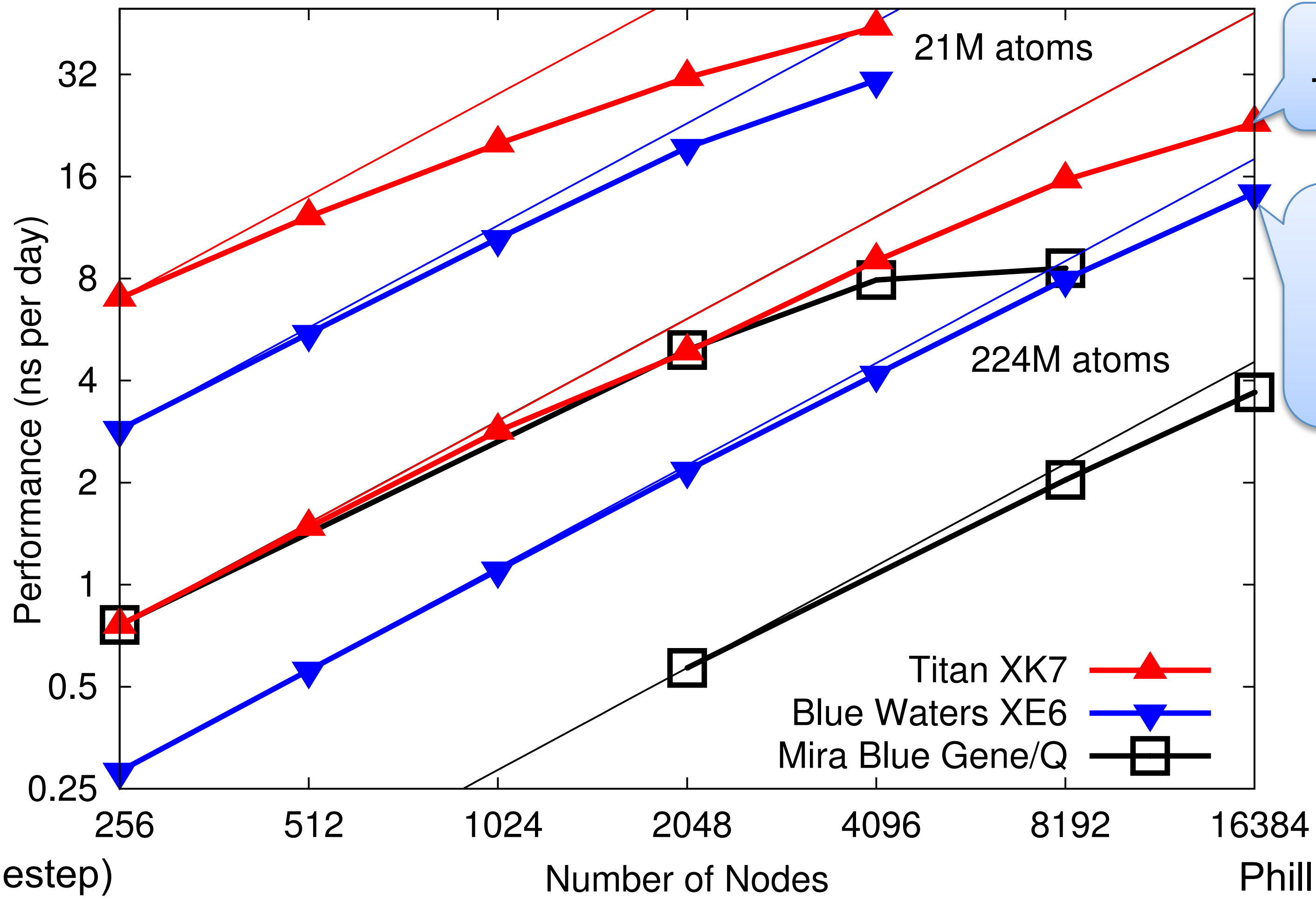
- 2006 - NSF calls for 100 million atom simulation
  - Had just published million-atom virus simulation
- Issues to address:
  - Find scientific questions worthy of resources
  - Build model and initial coordinates
  - Store output trajectory
  - Analyze output trajectory
  - Scale NAMD to 100 million atoms
  - Scale NAMD to petascale machine(s)

NAMD for Large Systems

- Per-node memory usage
  - Exploit redundant structure
  - Pre-compressed static data
  - Distributed per-atom data
  - Special “memopt” build
  - Not all features supported
  - NAMD-only file formats
    - May change between versions
    - No VMD reader/writer - ever

- I/O performance
  - Data is relatively small
  - Parallelized POSIX I/O
  - Performance is just OK
  - New Charm++ I/O library being co-developed
- Parallelize load balancer
  - Local load balancing only

NAMD on Petascale Platforms



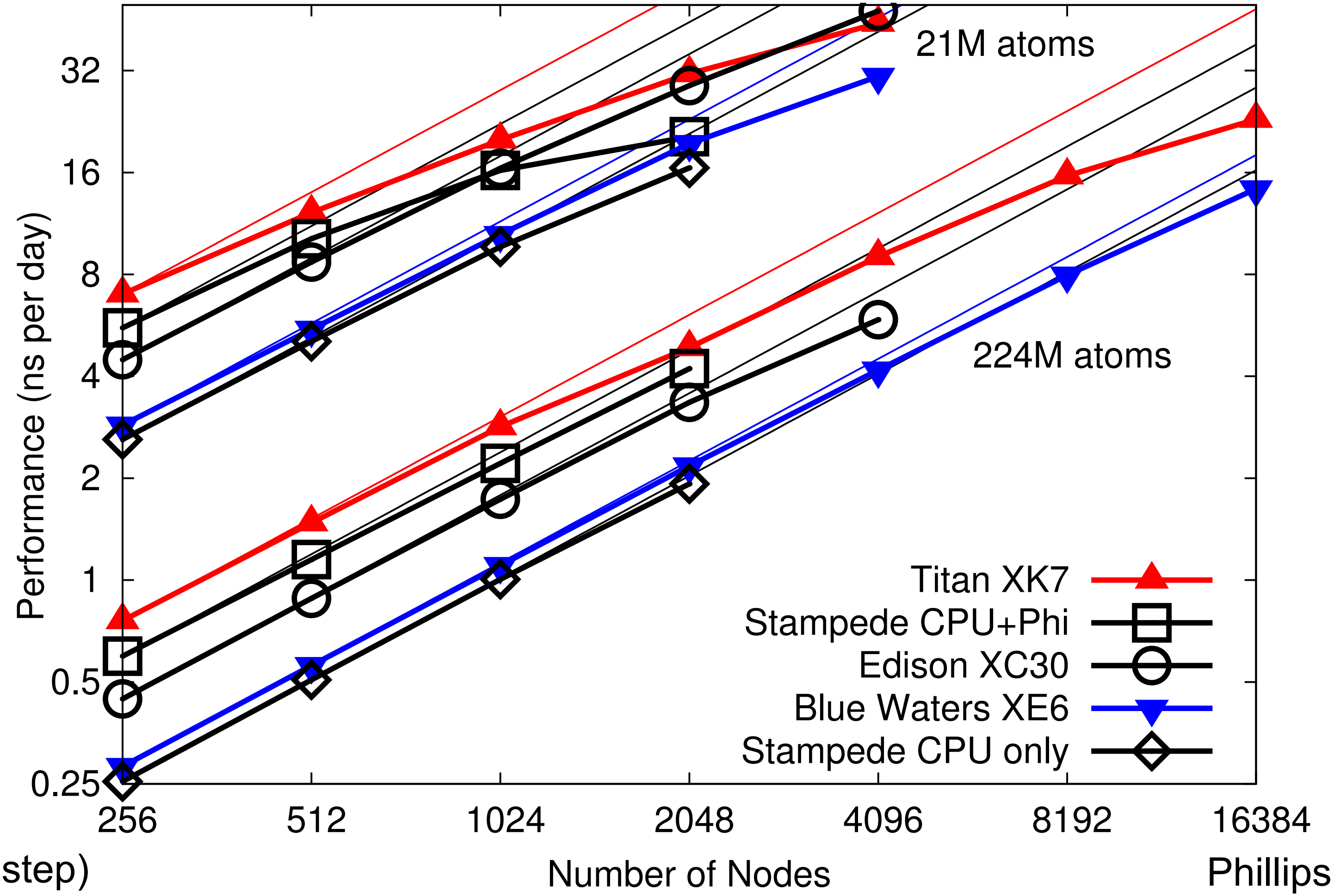
23 ns/day,
7.5 ms/step

14 ns/day,
79% parallel
efficiency on
224M atoms

(2fs timestep)

Phillips *et al.*, SC14

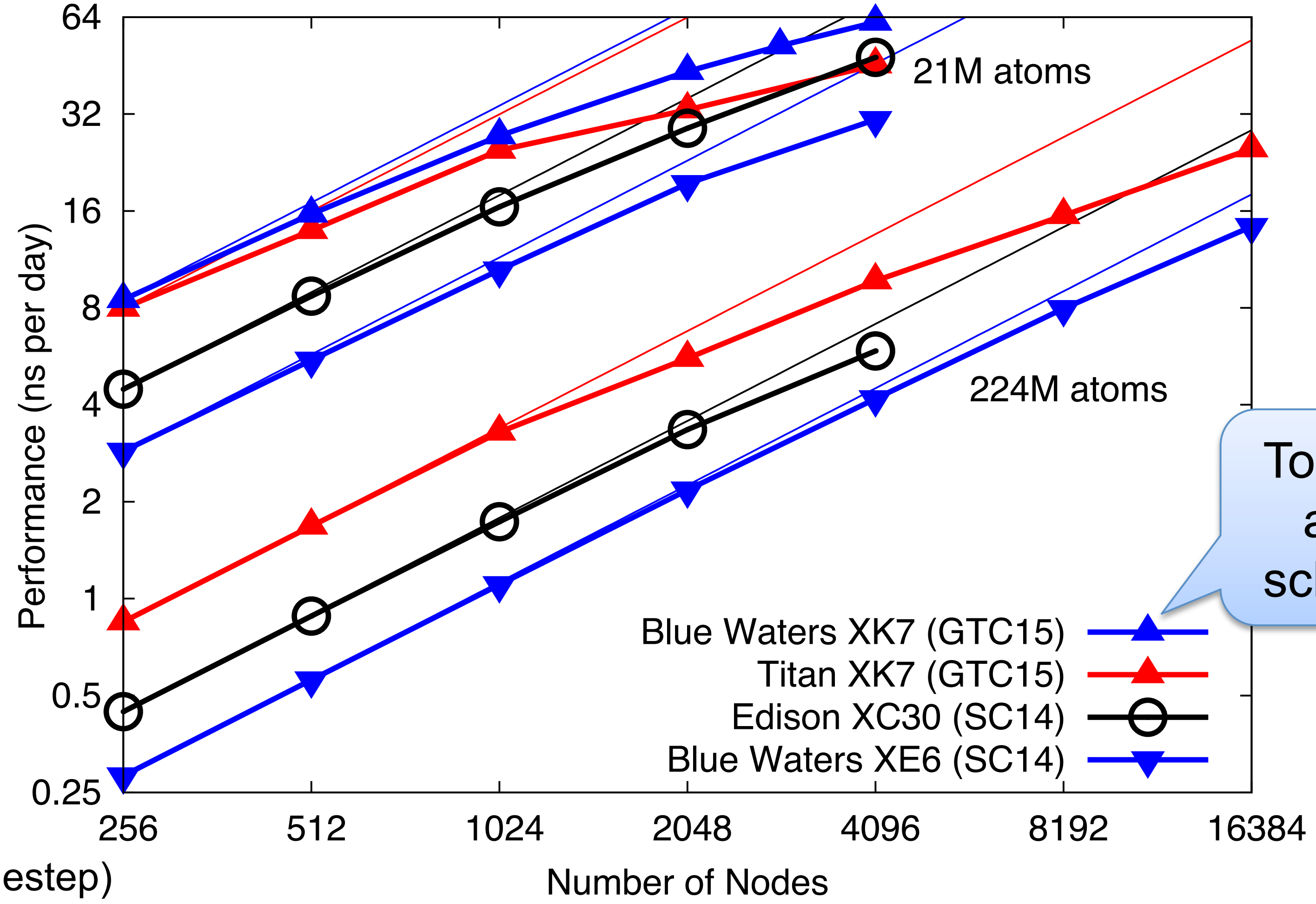
NAMD on Torus and Non-torus Networks



(2fs timestep)

Phillips *et al.*, SC14

NAMD on Petascale Platforms Today

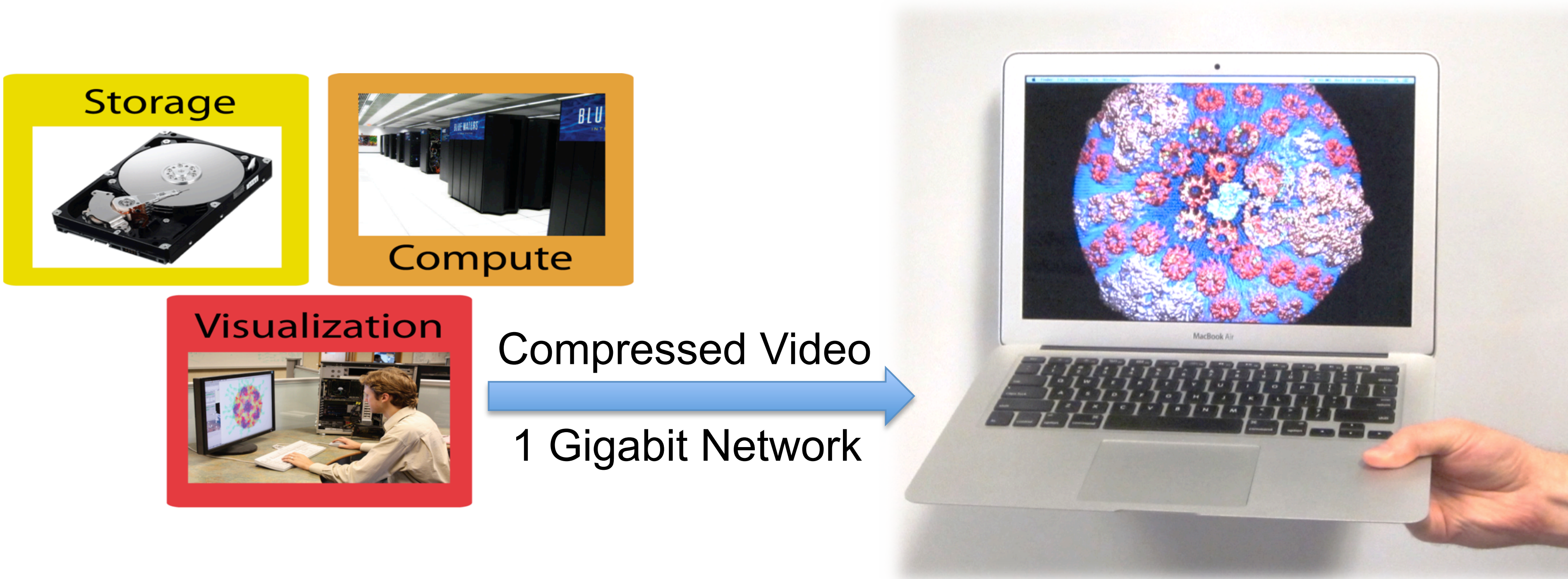


Topology-aware scheduler

(2fs timestep)

Remote Visualization Enables Petascale Anywhere

High-end visualization and analysis workstations previously available only in person at the Beckman Institute are now accessible via remote visualization.



Storage

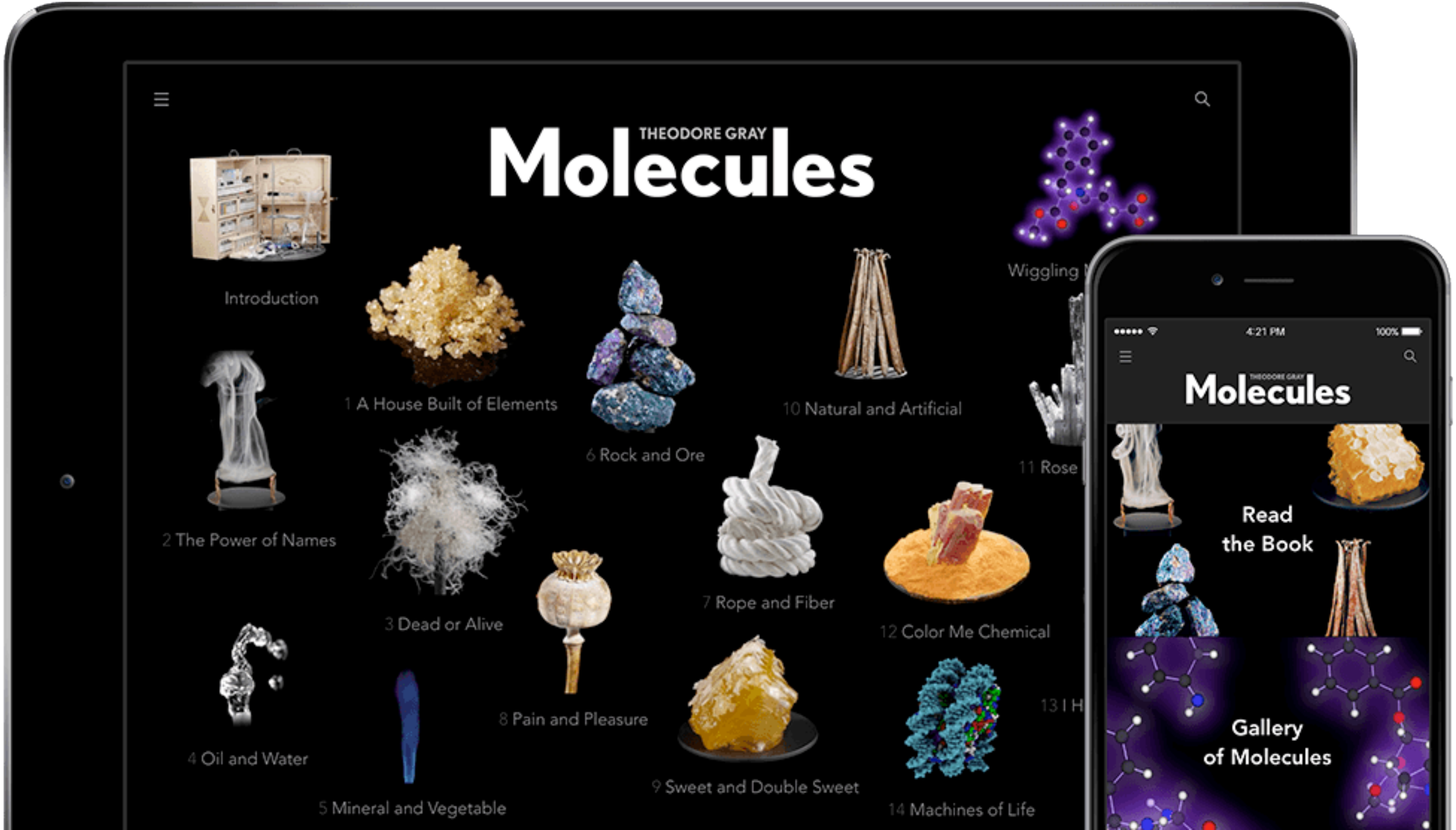
Compute

Visualization

Compressed Video

1 Gigabit Network

NAMD Runs on Smaller Platforms Too



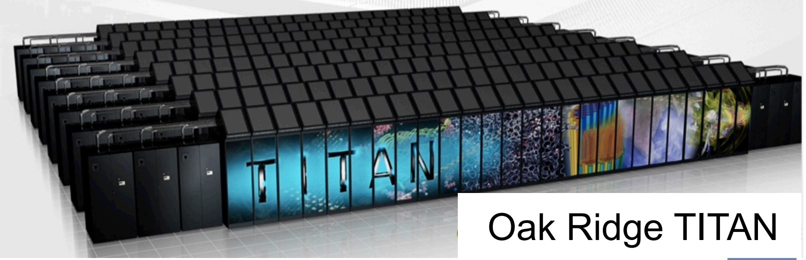
NAMD Mission Statement:

Practical Supercomputing for Biomedical Research

- 77,000 users can’t all be computer experts.
  - 18% are NIH-funded; many in other countries.
  - 23,000 have downloaded more than one version.
  - 5000 citations of NAMD reference papers.
- One program available on all platforms.
  - Desktops and laptops – setup and testing
  - Linux clusters – affordable local workhorses
  - Supercomputers – free allocations on XSEDE
  - Blue Waters – sustained petaflop/s performance
  - GPUs – from desktop to supercomputer
- User knowledge is preserved across platforms.
  - No change in input or output files.
  - Run any simulation on **any number of cores.**
- Available free of charge to all.

nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
THE HIV-1 CAPSID
Atomic structure of the AIDS pathogen’s protein coat
PAGE 643

Hands-On Workshops



Oak Ridge TITAN

Distribution and Licensing

- Binaries and source code
  - Charm++ included
- Annual releases
- Nightly builds
- Registration required
- Public CVS access available
- Installed on supercomputers

- No redistribution
- Citation required
- Registration required
- Use for any purpose
- Combine up to 10% of source with at least 50% original code without restriction
- VMD plugins use BSD license

Support and Training

- Public mailing list
  - Other scientists know best
  - Archived and searchable
  - Social conventions apply
- Bug report emails
- Personal support
  - Driving projects
  - New capabilities

- Tutorials and Case Studies
  - Written by scientists
  - Focus on science problems
- Hands-on workshops
  - Taught by scientists
  - Several per year
  - Various locations
  - Requires only laptop

Software Engineering

- Charm++:
  - Git revision control
  - Redmine issue tracking
  - Gerrit code review
  - Nightly build & test
- NAMD:
  - CVS revision control
  - Manual tracking & review
  - Nightly build & release
  - Automated build & install

- Configuration Management
  - "Latest" builds for users
  - Older builds preserved
  - Modules in use logged
  - Charm++ ships with NAMD
- Verification and Validation
  - New tests for new features
  - Library of historical tests
  - User reports of crashes
  - Internal checksums

Every Run is a Test

- Internal consistency checks are critical
  - Relatively few users, fewer developers
  - Variety of use cases and feature sets – runs are unique
  - Configuration changes are rampant on supercomputers
  - Testing at scale is expensive – leverage production runs
  - Crashes are annoying but harmless
  - Goal is to avoid generating bad science
  - Users do not read warning messages!
    - They barely read “FATAL ERROR” messages on stderr.
    - Exit on cases that risk incorrect results, link to web page.

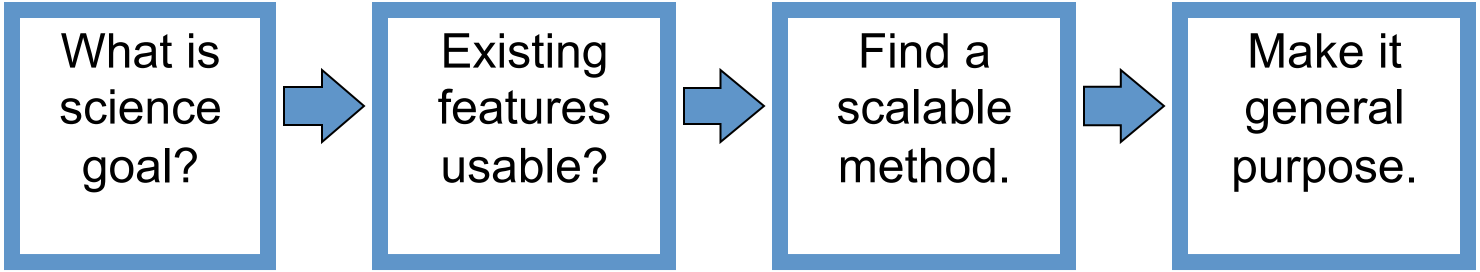
Development Process/Philosophy

- Five-year funding cycle
  - Code, science, publish, proposal
- Evolutionary development
  - Fully functional code at all times
  - No stable/development branches
  - Large changes by refactoring only
- Simplify – don’t manage
  - Separation of responsibilities
  - Alignment of incentives
  - Low coupling between people

- No code without an eager user
- No single-user features
- No schedules, no promises
- No design/code documentation
  - Source code must be **discoverable**
  - Use sandboxes to hide complexity
- Priorities and opportunities
  - Enabling new science
  - Supporting outside developers

Collaborative Driving Projects

- Nearly every experimental collaboration relies on NAMD.
- High-end simulations push scaling efforts.
  - Try to anticipate needs: Million-atom virus just worked in 2006.
- Innovative simulations generate feature requests:



What is science goal?

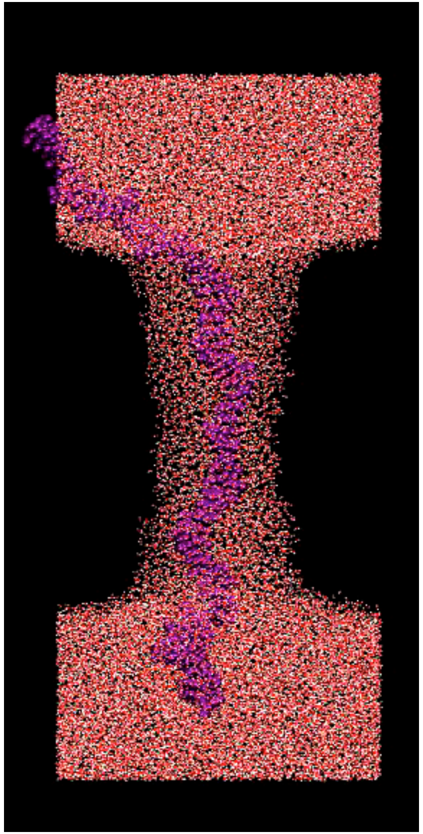
Existing features usable?

Find a scalable method.

Make it general purpose.

Portable Customization Through Scripting

- Top-level protocols:
  - Minimize, heat, equilibrate
  - Simulated annealing
  - Replica exchange (originally via sockets)
- Long-range forces on selected atoms
  - Torques and other steering forces
  - Adaptive bias free energy perturbation
  - Coupling to external coarse-grain model
- Special boundary forces
  - Applies potentially to every atom
  - Several optimizations for efficiency
  - Shrinking phantom pore for DNA

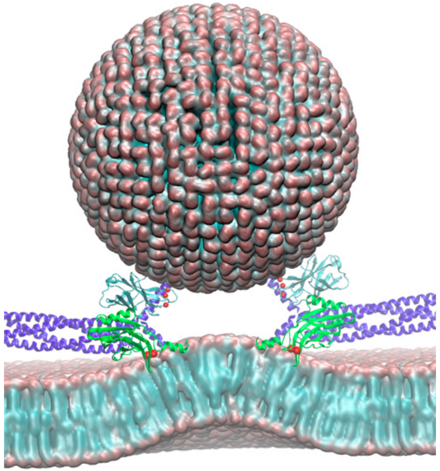


Why NAMD and VMD Use Tcl

- History: Programs are ~20 years old.
- Maturity: Package management, portable.
- Stability: Interfaces haven’t changed.
- Flexibility: Encapsulates mini-languages.
- Approachability: Looks like a simple scripting language, doesn’t scare non-programmers.

Looking Forward

- NERSC Cori / Argonne Theta (2016)
  - Knight's Landing (KNL) Xeon Phi
  - Single-socket nodes, Cray Aries network
  - Theta Early Science Project:
    "Free Energy Landscapes of
    Membrane Transport Proteins"
- Oak Ridge Summit (2018)
  - IBM Power 9 CPUs + NVIDIA Volta GPUs
  - 3,400 fat nodes, dual-rail InfiniBand network
  - CAAR Project "Molecular Machinery of the Brain"
- Argonne Aurora (2018)
  - Knight's Hill (KNH) Xeon Phi



Synaptic vesicle and
presynaptic membrane

Portability Requires Trade-Offs

What is “Portability”?

- *Minimal* Portability: Source code can compile and run correctly on multiple target platforms.
  - Not trivial. Lucky to work on multiple compilers.
- *Ideal* Portability: Near-ideal performance on target platforms (present and future) achieved without platform-specialized redundant implementations.
  - Not likely. Can’t even do this for matrix multiply.
- *Practical* Portability: Platform-specialized code is limited and compartmentalized.
  - Software engineering manages essential complexity.

Practical Portability Principles

- Use the right tools for each platform.
  - Non-portable performance is hard already.
  - Complexity is exponential, duplication is linear.
    - And you only need a single set of tests.
  - Platforms evolve two ways:
    - Divergent – duplication simplifies maintenance
    - Convergent – refactor to remove redundancy

Practical Portability Principles

- Do the job right for each platform.
  - Optimize based on feedback, not intuition.
    - Profilers exist because intuition fails us.
    - Try multiple strategies – automate if possible.
  - Don’t abandon software engineering.
    - Redundant code is a strategy, not a sin.
    - *Essential* complexity can only be managed.

Practical Portability Principles

- Don’t predict the future – plan for change.
  - Focus on the problems you have now.
  - Adapt to change as it happens.
- General solutions don’t just happen.
  - They are generalized from specific cases.
  - There is no substitute for experience.
  - Do eliminate redundancy where you can.

Case Study: Charm++ Network Layers

- Circa 2000
  - Linux, Linux-Alpha, AIX, HP-UX, IRIX, Tru64, Solaris, Mac, Win32
  - net, net-tcp, MPI, Origin2000, T3E (shmem)
- Today
  - Linux, Linux-Power, Linux-ARM, Mac, Win32
  - Low-level Runtime System (LRTS) removes redundant code:
    - multicore, netlrts, verbs, gni, pamilrts, MPI
  - Legacy layers: net, net-tcp, net-ibverbs, pami, …
  - New features (e.g., replica partitions) only in LRTS

What to Expect

- Likely common (count your blessings):
  - Inter-node parallelization
  - High-level algorithms
  - Anything not worth tuning

- Likely specialized:
  - Kernels (obviously)
  - Work scheduling
  - Data layout

Aspects of Portability in NAMD

- Operating systems
  - Linux, Mac, Windows
  - Lustre filesystem errors
  - System library and OS bugs
- Networks
  - Infiniband, Gemini, BG/Q
  - Offload to MPI/Charm++
  - Do not use Charm++ on MPI
  - Charm++ relatively fast to port
- CPU architecture
  - Compiler directives (e.g., ivdep)
  - OpenMP 4.0 “#pragma omp simd”
  - Occasional vector intrinsics
- Coprocessors
  - CUDA is mature and best in class
  - OpenCL isn’t performance-portable
  - OpenACC supported by Intel?
  - Intel offload directives only for MIC

Offload Challenges

- Writing CUDA kernels is acceptable
  - Vendor is fully engaged
  - Knowledge is widespread
  - Tools are mature *and* up-to-date
- Offload aggregation is the challenge
  - Charm++ has multi-threaded control on CPU
    - Independent threads that share a memory space
  - GPU works best with unified stream of work
- Work remaining on CPU will become bottleneck
  - Can stream results off GPU to enable overlap
  - Need to optimize, use all CPU cores/threads available

Charm++ and MIC Options

- Today: Aggregated Offload
  - Keep NAMD work decomposition
  - Collect and bulk-copy data
  - Bulk-launch tasks in single offload
  - Method initially developed for CUDA
  - NAMD MIC offload is clone of CUDA offload

Charm++ and MIC Options

- KNL Option: OpenMP Thread Teams
  - Grainsize too large for single MIC thread
  - Grainsize too small for entire MIC
  - Let Charm++ control OpenMP thread teams
    - E.g, MIC = 15 Charm++ PEs
    - Each PE = 4 cores and 16 threads
    - Parallelize loops using OpenMP directives

Code Modernization

- Re-vectorization? Post-modernization?
- NAMD is currently array-of-structures
  - Aligned to cache lines for random access
- Emerging idiom for MD kernels:
  - Load atoms in neighbor list from AoS
  - Transpose in register for vectorization
  - Gather-scatter may provide similar performance

MIC Vectorization Options

- Intrinsics – what we have
  - Written by David Kunzman (formerly) of Intel
  - Currently in production
- Compiler – what we want
  - #pragma [omp] simd assert
  - In 2014 ~20% slower with refactored kernel
    - Missed AoS optimization now available in compiler
- ISPC – our backup plan
  - Similar to CUDA, consider if compiler fails

Conclusions and Ramblings

- Science, software, and supercomputing are all hard.
  - If you get good science from any supercomputer, you are winning.
- Solve problems you have before problems you might have.
  - Performance and correctness on one platform, then portability.
  - Complexity is forever - try the simplest thing that might work.
- Do look ahead - don't paint yourself into a corner.
  - But don't worry about things you don't yet understand well.
  - If you do get stuck, don't be afraid to refactor.
- If a problem has many solutions, it is probably unsolved.
  - But even a limited tool may work for your case.

Thanks to NIH, NSF, DOE, and 20 years of NAMD and Charm++ developers and users.

Want to learn more while you're at SC15?

- Chemical Visualization of Human Pathogens: the Retroviral Capsids
  - *Finalist, SC15 Visualization and Data Analytics Showcase*
  - Wed. Nov. 18, Ballroom E, 10:30am-12:00pm

- *Charm++ and AMPI: Adaptive and Asynchronous Parallel Programming*
  - Thu. Nov. 19, Room 13B, 12:15-1:15pm

- NAMD: Innovation Beyond Petascale
  - Fri. Nov. 20, Hilton 408, 9-9:30am, *Molecular Simulation Software Workshop*

- Full list of all NAMD/VMD/Charm++ events:

http://www.ks.uiuc.edu/events/sc2015/